DATA AUTOMATION DEVELOPMENT AND SYSTEMS IMPLEMENTATION-- SOME PROBLEMS AND CONCLUSIONS

A. E. Wessel

August 1967

P-3648

DATA AUTOMATION DEVELOPMENT AND SYSTEMS IMPLEMENTATION-- SOME PROBLEMS AND CONCLUSIONS

A. E. Wessel*

The RAND Corporation, Santa Monica, California

Since the early 1950's, I have been involved, as have others here today, in the various phases of data automation system development. This has included requirements definition, system design, hardware and software development, and system implementation and modification. During this same period of time, just as many of you, I have been involved as an actual, but usually unhappy, systems user. Being both the system builder and user has at least one interesting consequence. It is not quite so easy to blame the other fellow for your own difficulties. You are the other fellow! Therefore, there is a strong motivation to ask some questions concerning what went wrong and what need not and should not go wrong in the future and an equally strong motivation to get some useful answers to such questions.

The intent of this talk, therefore, is to share with you some thoughts I have had and some tentative conclusions I have reached

---

*Any views expressed in this paper are those of the author. They should not be interpreted as reflecting the views of The RAND Corporation or the official opinion or policy of any of its governmental or private research sponsors. Papers are reproduced by The RAND Corporation as a courtesy to members of its staff.

This paper was prepared for presentation at a meeting of the International Patent Officer's Association (ICIREPAT), Stockholm, Sweden, September 18-23, 1967.

concerning the development and implementation of data automation or information processing systems over the past decade. In addition, I want to tell you a little bit about an on-going experimental patent search system being implemented by the research and development people of the U.S. Patent Office in conjunction with The RAND Corporation. As I was the system designer of the functional software (the hardware and internal software already existed as the RAND JOSS* system which I used to produce the functional software**), we can consider this experimental system a case in point in applying some of the more general conclusions I will now discuss.

One of the things I have learned is that while all data automation or information processing systems have some common characteristics, there are some highly critical differences among them as a result of the various uses for which they are intended. For example, if we are concerned with building an information processing system for business accounting, payroll, or billing, an efficient batch processing system with strict procedural and administrative rules with regard to use, access, data entry and output, data processing requests, etc., is likely to provide an efficient operational system. On the other hand if we are concerned with providing an information processing system for man-machine interaction in problem solving, planning, or decisionmaking (and I regard search and retrieval systems for patent examination as being of this kind), it is almost certain that such an

---

*JOSS is the trademark and service mark of The RAND Corporation for its computer program and services using that program.

**With the aid of Carroll Lindholm, The RAND Corporation.

efficient batch processing system with the usual administrative restrictions will result in a very discontented group of system users.

There are many reasons for drawing this conclusion, the chief of which is simply that the procedures, logic, data requirements, and techniques involved in solving problems and making inferences, plans, and decisions remain somewhat murky, and that any given clarification and detailed analysis, if fixed upon for data automation system design, will result in a system too inflexible for efficient operational use.

Let us call this latter category of system application one of "interactive problem solving operations." While it is difficult to describe this category succinctly, I shall mean this term to cover systems that support a wide variety of people involved in making inferences, decisions, and plans, as well as solving problems. Usually such systems are also intended to support people involved in conveying to someone else the results of these activities. Furthermore, it includes information processing support to those human activities that involve implementation of plans, decisions, etc., and the monitoring of the results of the attempted implementation.

For contrast, let us call the former category of system application one of "administrative data flow." This term is intended to cover such applications as payroll, accounting, insurance claims, and the like.

There is a third possible type of information processing system, which we can call "control" systems. This type is used in such activities as oil processing and refining, fire control of weapon

systems, vectoring of aircraft to targets or runways as in various automated landing systems, and so forth.

Now I have claimed that there are some critical differences among these various categories of information processing systems, and I have given one very general example of what the difference can imply. While I do not believe this to be a very controversial claim anymore, it turns out that in the real world any given existing data system implementation is more likely to be a mixture of various categories (perhaps including some I've not mentioned) than to be solely applied to just one of these categories. Most "control" systems, for example, provide data for management administration and management problem solving, decisionmaking and planning. Furthermore, the typical data flow in existing automated systems works downward as well as upward and laterally.

The point of all this is that the actual situation in information system applications is bad. I mean by this that the existing mixed system, which often uses the same hardware and batch processing for different purposes, is a bad one from the point of view of the needs of the problem solving, inference and decisionmaking community of users, where interaction between man and machine is essential if computers are going to be of any significant help.

I wish to expand a little on this point. Without going into the detailed case histories*, let me simply offer you some conclusions.

---

*See Some Thoughts on Developing Future Command and Control Systems, The RAND Corporation, P-2941-1, S. M. Genensky, A. E. Wessel, October 1964, and The Impact of the New Technology on Command System Design, The RAND Corporation, P-3409, A. E. Wessel, July 1966.

The severe difficulties involved in obtaining realistic operational requirements for systems to be applied to interactive problem solving operations lie at the heart of the difference between such systems and those to be applied to administrative data flow or control operations. Even with significant user participation, it has not been possible for the system developer, using the usual systems analysis, to obtain a detailed description of users' needs and operational requirements that could be translated into a coherent functional design and satisfactorily guide the system designer in the long-term development of such systems. Both the military and civilian experience with which we are familiar has highlighted the serious difficulties that exist in trying to obtain the detailed descriptions of the current operations for which data automation support is to be provided, and, therefore, the functional requirements for information processing. Furthermore, even assuming the partially successful specification of the operational requirements (whether accomplished by the user, by some outside agency, or by an association of both), it is important to recognize that any such description is of necessity not invariant with respect to time. Thus, it is clear that before a major system development that might satisfy the specified operational requirements could be completed, changes in policy, practice, user expectations, and needs would have outmoded the developmental system.

While some means have been found whereby the potential system user could introduce up-dated functional requirements that could result in program-design changes, compromise is inevitable. In order that the established delivery schedules might be met, great pressure

is applied to freeze the system design early so that hardware procurement and development can begin. Soon after the selection of the contractors there is a tendency to fix upon an initial operational capability, reserving changes for the second phase of some specified operational capability. It is then quickly discovered that most aspects of the system, both hardware and software, soon became cast in concrete. At this point, introduction of even relatively small program-design changes, however important, become costly in terms of both funding and delivery time. This experience has been the common pattern, with the potential user often finding that a major computer reprogramming effort is required before the delivered system can become operational.

In the light of these difficulties, some of us have sought to develop general-purpose data processing equipment and computer programming. This technological effort has been directed towards the development of equipment and associated software packages flexible enough to permit a resultant system to be transformed by the user as his needs change and become better known through operational experience. Whatever future technological development may bring, it remains the case that current equipment and computer programming techniques continue to impose severe difficulties upon the system user. A major time- and manpower-consuming effort is still required in order for the user to achieve an operationally useful system given the delivered hardware and software packages.

In the light of all this it should be of little surprise to us that, while there has been and still is a plethora of talk concerning

the potential of information processing systems as applied to decision-making, problem solving and planning activities, there has been little realization of such systems. The system developers and hardware manufactures talk decision theory but they implement administrative data flow and control systems. All of which, true enough, provides some indirect support to the problem solvers and decision makers but fails to come to grips with the basic problem of providing direct computer interaction for the people engaged in these higher level activities. This is of special importance to all of us here today because I am convinced that search and retrieval for patent examination involves such higher level activities. The question before us then is whether we will continue to be offered only talk of data automation for interactive problem solving operations and data automation implementation only of administrative data flow systems.

Having raised this issue, it is only fair to conclude my talk with a concrete example of one experimental system that I believe does attempt to provide such interactive support for patent search activities. This example is presented not because I believe it to be the be all and end all of such systems but because it is illustrative of the kind of system design and implementation I believe necessary to provid direct support to those engaged in such activities as patent search and examination.

First let us consider the approach taken. As we all know, perhaps the fundamental problem that confronts us in patent search and retrieval systems is the data classification, indexing, or description problem. The workers in the fields of library science, machine translation, and

fact correlation have labored long and mightily for more than ten years in the vineyards of government-sponsored research on the overall problem and, to date, have produced little.* At the same time we know that a workable, though tentative, set of index terms is constructable for any given specific patent field. In the design of the experimental system, I therefore assumed that we could achieve a tentative list of index terms, and in conjunction with Walter Burns, Staff Director of Research, U.S. Department of Commerce, Patent Office, Washington, D. C., we selected the specific field of fluidics. Furthermore, I assumed that the initial index list would be changed both during the experiment and in actual operational use by patent examiners. A second set of assumptions had to do with the search processes to be permitted. Basically this meant that I wanted to permit any kind of search request (even illogical ones) based on the tentative and changing index list. One should be able to ask for the index terms associated with patents, the patents associated with any one index term or any set of up to twelve index terms with any ordering of "ands," "ors," and/or "nots." A third set of assumptions involved the input process. I wanted it to be easy. I made the deletion process difficult because I did not want someone to erase my files during the experimental phase during which I did not want to impose restrictive rules of access. I then assumed that the patent examiners would not be computer programmers and would not wish to go through a programmer or punch card middleman

*See Publication 1416, National Academy of Sciences, National Research Council, Washington, D. C.

to make inputs, deletions, or searches. I chose to give the examiners free, direct, and easy access to the computer files. I also assumed that the examiner would make all kinds of mistakes. Therefore, I wanted to make it easy for him to make mistakes if he wanted to, difficult to make unintended mistakes, and, in any event, easy to correct whatever mistakes were made. Finally, I assumed that any system I set up would change, requiring reprogramming during the experiment. Therefore, I wanted reprogramming to be easy.

Before describing this experimental system, a few words about the JOSS system itself might be helpful.* JOSS is an on-line, time-shared system that permits many different users in RAND, Santa Monica, California, and at certain locations in Washington, D. C., to have direct access to the PDP-6 computer at RAND, Santa Monica. The JOSS internal machine language is usually not available to the user. The user is given a "metalanguage" consisting of conventional English language direct and subjunctive commands. In less than one-half hour, the ordinary user can learn the possible list of commands and begin to converse with JOSS. Other than this JOSS-user language, the only component of the system that the user is aware of is an IBM Selectric Typewriter** specially modified for RAND. The user commands are typed out in green; the JOSS response in black. The JOSS user is not only able to give JOSS direct commands; he is also able to

---

*For a more complete description of JOSS, see JOSS: Introduction to a Helpful Assistant, The RAND Corporation, RM-5058-PR, C. L. Baker, July 1966, and JOSS: Rubrics, The RAND Corporation, P-3560, C. L. Baker, March 1967.

**A few special cases use a teletype console.

produce rather elaborate programs or instructions. Appendix B, which presents the basic JOSS programs for the experimental patent search system, is an example of the JOSS program-building capability. Those of you interested may compare this kind of program with those produced using, say, FORTRAN. Note, however, that the nonprogrammer-oriented language is not the only benefit of JOSS. The JOSS user programs are produced on-line with direct interaction with the JOSS computer. The user receives significant aid in this manner, including a certain amount of direct or immediate error checking and diagnosis. The program in Appendix B was produced in two days and later revised (in less than one day) to incorporate some wanted improvements. I will have more to say concerning this point later; however, the ease of progr revision is one of the features I wanted for the experimental system. It is built for change.

Appendix A presents the experimental patent search system's instruction manual. Note its length, 4 pages, and its simplicity. In effect, I have created a "meta-metalanguage" for the patent examiner's use, permitting him to converse with the JOSS computer and to input, delete, or search for data. To do this, the examiner has to learn to use the following ten English commands, including one "subjunctive" search expression:

1. Use file_____.	(This calls up the patent file. It is coded so that only the appropriate examiners can reach the file.)
2. Recall item 1 (INPUT).	(This calls up the necessary JOSS program for adding patents and/or index terms to the file.)

2a. Do part 1.	(This activates the input program.)
3. Recall item 2 (DELET).*	(This calls up the necessary JOSS program for deleting index terms associated with patents.)
3a. Do part 2.	(This activates the deletion program.)
4. Recall item 3 (SERCH).	(This calls up the JOSS program for searching by patent or by index terms.)
4a. Do part 3.	(This activates the search program.)
5. Let T = (up to 12 variables connected with logical "ors," "ands," or "nots.")	(This requests the more complex search by up to 12 index terms.)
5a. Do part 4.	(This activates the logical search program.)
6. Delete all.	(This empties the working memory, but not the file, of data and programs extraneous to the next task.)

Rather than go into more detail concerning these commands let us see how they and the experimental patent search system function. The best way to demonstrate this experimental patent search system would be to have made available a JOSS console for you to try for yourself with no further instructions than those offered in Appendix A. While that is impractical today, I expect it would be a feasible thing to do in the next few years,** and we might just note that fact and its implications for international or regional systems of the future. The next best way to give you some understanding of the experimental system is to go over the example of the "examiner"/JOSS conversation presented below:

---

*Current JOSS codes for files and items in files are restricted to no more than five alphanumerics.

**I do not necessarily mean the actual JOSS system here but some similar time-sharing system of the future.

JOSS at your service.
Initials please: aew
Prcject number: 7115
Department: elec

```
Use file 174 (elec4).
Roger.
Recall item 1.
I can't find the required item.
Recall item 1 (input).
Done.
Do part1.
Eh?
Do part 1.
This part is used for input to the file.
Set patent number and index term to zero to conclude input.

Patent Number = 123456
Index Term = 12
Index Term = 34
Index Term = 0
Patent Number = 345678
Index Term = 14
Index Term = 34
Index Term = 12
Index Term = 10
Index Term = 23
Index Term = 0
Patent Number = 0
Delete all.
Recall item 2 (delet).
Done.
Do part 2.
This part deletes items from the file.

Patent Number = 345667*
Patent Number = 345678

Index term to be deleted = 23
Index term to be deleted = 0

Patent Number = 123456

Index term to be deleted = -250
All items will be removed.

Patent Number = 0
Delete all.
Recall item 3 (serch).
Done.
Do part 3.
This is the file search phase.
A patent number or index term set to zero concludes that part.
Part 1 finds index terms associated with a given patent number.
Part 2 finds patents having a given index term.
Part 3 finds patents according to logical combinations of
index terms.

I want part = 1
```

```
Patent Number = 123456
Patent number   123456 has the following index terms:
No terms found.

Patent Number = 345678
Patent number   345678 has the following index terms:
  10
  12
  14
  34

Patent Number = 123457
Patent number   123457 has the following index terms:
  12
  14
  15
  17
  18

Patent Number = 678954
This patent number not in file.
Patent Number = 1
Patent number        1 has the following index terms:
   1
   2
   3
   4
   5

Patent Number = 0
I want part  = 2
Index Term = 12
The following patents refer to index term   12:
  123457
  123458
  123460
  345678
Index Term = 78
The following patents refer to index term   78:
No entries found for this index term.
Index Term = 13
The following patents refer to index term   17:
  123457
  123459
  123460
Index Term = 0
I want part  = 3

On next line type a search expression.  Then type: Do part 4.

Let T = (a or b) and not c.
Do part 4.
Number of variables used in above expression =
For each variable give the corresponding index term.
```

```
a = 12
b = 10
c = 19
The following patents have been located:
  123457
  123460
  345678
Search complete.

Enter a new search expression if you wish. Then do part 4.

Let T = (a and b) and not c.
Do part 4.
Number of variables used in above expression = 3
For each variable give the corresponding index term.
a = 12
b = 10
c = 19
The following patents have been located:
  345678
Search complete.

Enter a new search expression if you wish. Then do part 4.

Let T = a and not a.
Do part 4.
Number of variables used in above expression = 1
For each variable give the corresponding index term.
a = 10
No entries found for this search.
Search complete.

Enter a new search expression if you wish. Then do part 4.

Let T = a or not a.
Do part 4.
Number of variables used in above expression = 1
For each variable give the corresponding index term.
a = 10
The following patents have been located:
       1
       2
       3
  123456
  123457
  123458
  123459
  123460
  213456
  213457
  213458
  243777
  345678
Search complete.
```

Enter a new search expression if you wish. Then do part 4.

Let T = a.
Do part 4.
Number of variables used in above expression = 1
For each variable give the corresponding index term.
a = 12.
The following patents have been located:
123457
123458
123460
345678
Search complete.

Enter a new search expression if you wish. Then do part 4.

Let T =
Eh?
Let T = b.
Do part 4.
Number of variables used in above expression = 1
For each variable give the corresponding index term.
a = 12
The following patents have been located:
345678
Search complete.

Enter a new search expression if you wish. Then do part 4.

Let T = a.
Do part 4.
Number of variables used in above expression = 1
For each variable give the corresponding index term.
a = 10
The following patents have been located:
345678
Search complete.

Enter a new search expression if you wish. Then do part 4.

Delete all.

Type users.
users: 7

I think I can now give a qualified but positive answer to the question raised earlier; namely, can we expect direct data automation support for interactive problem-solving operations in general and for the patent examiner search activities in particular? My experience with the experimental system I have just discussed with you has been, admittedly, brief. We are just now entering real patents and real index terms into the system. Certainly a progress report one year from now would be appropriate. But even at this early date in the experiment, I feel confident that direct data automation support to the patent examiner permitting him to interact with the computer is not only feasible but actually available in the admittedly limited manner described in the experiment. My qualifications of that positive statement are these. One still requires the index list, but one need not solve the whole data classification problem. We expect and encourage change and improvement of the index list within the experiment. In fact the building of tentative and evolving index lists is one of the reasons for the experiment. Another point to be made is that we should not rest content with the development of "administrative data flow" systems. We have to demand direct and interactive computer support for the patent examination process. If we do not make this demand, the system developers and computer manufacturers will take the easier course of only automating administrative data flow. My last qualification has to do with the need to insure free and easy access to the system for the patent examiner. Let us not be sold on so-called "efficient" systems which involve the patent examiner in administrative red tape before he can get to interact with the system. Of course,

we must protect the system files, but the system can provide "individual" files as well as system files. Of course, we must expect the patent examiner to be willing to learn a few rules in order to use the system, but we need not expect him to learn how to program computers or to learn a complicated new language. To sum this all up in one phrase, we not only have to demand direct and interactive support for the patent examiners, but we also have to demand the right kind of system. I have attempted to indicate to you today some of the more important characteristics of what I believe to be that right kind of system.

Appendix A

EXPERIMENTAL PATENT SEARCH SYSTEM INSTRUCTION MANUAL

Carroll R. Lindholm

This is a users' guide to the programs now written to do a very un-JOSS-like job on the JOSS console. Many awkwardnesses will be evident where I have tried to squeeze a file-retrieval-program foot into the JOSS-language shoe. However, JOSS has consoles and numerical and logical commands so the task was attempted.

Input of Information. Before the program can work on retrieval there must be some material to retrieve. Part 1 inputs material, either for the first time or to add new material. Part 2 permits deletion of material. The "material" consists of "patents" represented internally by their patent numbers and "index terms" represented internally also by numbers. Naturally alphanumeric characters would be preferable to numbers in an actual application but JOSS can only handle numbers. To input information do the following sequence of steps:

	Delete all.
	Use file 174 (ELEC4).
JOSS answers:	Roger.
	Recall item 1 (INPUT).
JOSS answers:	Done.
	Do part 1.

JOSS now requests a patent number and associated index terms.

To delete material:

Delete all.

Use file 174 (ELEC4).

JOSS answers: Roger.

Recall item 2 (DELET).

JOSS answers: Done.

Do part 2.

An index term of -250 will delete all items associated with a patent.

Part 2 is for deletion of index terms by patent number.

Retrieval. There are three retrieval schemes provided. All retrieval is done by:

Delete all.

Use file 174 (ELEC4).

JOSS answers: Roger.

Recall item 3 (SERCH).

JOSS answers: Done.

Do part 3.

Instructions on the console guide the user to the three subprograms. The first, obtained by typing "1," when requested merely types a list of those index terms associated with a given patent. Conversely, a "2" results in a list of patents possessing one given index term.

The third part is much more general and searches out those patents meeting a prescribed description put in by the user. At this point a "Search Expression" is required. This is always of the form:

Let T = logical expression.

The logical expression uses consecutive lower case alphabetic variables (a thru l) and logical connectives. For example: (a and b) or (not c and d). The letters a, b, c, d are to be identified with index terms and in the example will cause all patents having both index term a and b or having not c as w ll as d to be typed out. Examples below will help the reader formulate the logical expression. Again, because of the limitations of the JOSS language certain awkward characteristics are present. In particular the letters used in the logical expression must begin with a and be consecutive thereafter. When possible the computer directs the user. A value of zero is used to conclude a part. If any internal JOSS error messages occur the best solution is to Delete all and Recall the desired item and begin again. If valid values are given at each step, there should be no problem. Patent Numbers can be one to eight digits, index terms must be in the range -250 to +249.

Examples of Logical Expressions:

Let T = a. (Note the period at the end of the sentence.)

This retrieves those patents having an index term later identified as a. The command "Do part 4" activates such searches. Note th. t Let T = b or Let T = c are illegal expressions since the letters used must always include a and consecutive letter thereafter.

Let T = a and b. or Let T = a or b.

This illustrates the use of the simple logical "and" (meaning both a and b) and logical "or" (meaning a or b or both).

Let T = a and (b or not c).

Here those patents which do have index term a are desired. But among them, only those which also either have b or specifically lack c.

From the above examples, it can be seen that rather complicated searches are possible. Parentheses and brackets may be used (in pairs) as desired:

Let T = not [a and (not b or c or (not d) and not e) or f] and g.

Appendix B

EXPERIMENTAL PATENT SEARCH SYSTEM JOSS PROGRAM

```
Use file 174 (elec4).
Roger.
Recall item 1 (input).
Done.
Type all.

1.1 Type "This part is used for input to the file.".
1.11 Set Z=0.
1.2 Type "Set patent number and index term to zero to conclude input.".
1.21 Line.
1.22 Do part 11 for 0=10.
1.23 Let P be sparse.
1.24 Demand a as "Patent Number".
1.25 To step 1.8 if a=0.
1.26 Do part 101.
1.27 Done if j=250.
1.28 Set z=1+ip[(250+j)/100].
1.29 Do part 12 if z≠Z.
1.30 Demand i as "Index Term".
1.31 To step 1.24 if i=0.
1.32 To step 1.90 if |i|≥250.
1.33 Set A(i,j)=a.
1.36 To step 1.30.
1.8 Discard item 10.
1.805 File P as item 10.
1.81 Do part 14.
1.82 Done.
1.90 Type "Index terms must be less than 250.".
1.91 To step 1.30.

11  Recall item 0.

12.0 Use file 173 (ELEC3).
12.1 Do part 13 if 1≤Z≤5.
12.2 Delete A if Z≠0.
12.3 Recall item z.
12.4 Use file 174 (ELEC4).

13.1 Discard item Z.
13.2 File Z,A as item Z.

14  Use file 173 (ELEC3).
14.1 Do part 13 if 1≤Z≤5.
14.2 Use file 174 (ELEC4).

101.1 Set j=-250.
101.2 Set P(j)=a if P(j)=0.
101.25 Done if P(j)=a.
101.3 Set j=j+1.
101.4 To step 101.2 if j<250.
101.5 Done if j<250.
101.6 Type "Sorry, 500 patents have already been filed.".
```

```
Delete all.
Recall item 2 (delet).
Done.
Type all.

2.01 Do part 21 for 0=10.
2.02 Set Z=0.
2.05 Let P be sparse.
2.06 Type "This part deletes items from the file.".
2.07 Line.
2.10 Demand a as "Patent Number".
2.11 To step 2.90 if a=0.
2.12 Set P(250)=a.
2.20 Set j=first(z=-250(1)250:P(z)=a).
2.21 To step 2.8 if j=250.
2.25 Line.
2.26 Set z=1+ip[(250+j)/100].
2.27 Do part 22 if z≠Z.
2.30 Demand i as "Index term to be deleted".
2.31 To step 2.07 if i=0.
2.315 To part 25 if i=-250.
2.32 To step 2.92 if |i|≥250.
2.50 Type "Index term not present." if A(i,j)=0.
2.60 Delete A(i,j) if A(i,j)≠0.
2.61 To step 2.30.
2.80 Type "This patent number not in the file. Please enter the next one.".
2.81 To step 2.07.
2.90 Do part 24.
2.91 Done.
2.92 Type "Index terms must be less than 250.".
2.93 To step 2.30.

21 Recall item 0.

22 Use file 173 (ELEC3).
22.1 Do part 23 if 1≤Z≤5.
22.2 Delete A if Z≠0.
22.3 Recall item z.
22.35 Let A be sparse.
22.4 Use file 174 (ELEC4).

23.1  Discard item Z.
23.2  File Z,A as item Z.

24  Use file 173 (ELEC3).
24.1 Do part 23 if 1≤Z≤5.
24.2 Use file 174 (ELEC4).

25.0 Type "All items will be removed.".
25.1 Set P(j)=0.
25.2 Do part 22 if z≠Z.
25.3 Do part 26 for i=(-250)(1)250.
25.4 Do part 24.
25.5 To step 2.07.
```

```
26.0 Delete A(i,j) if A(i,j)≠0.


Delete all.
Recall item 3 (serch).
Done.
Type all.

3.1 Type "This is the file search phase.".
3.11 Type "A patent number or index term set to zero concludes that part.".
3.30 Type "Part 1 finds index terms associated with a given patent number.".
3.31 Type "Part 2 finds patents having a given index term.".
3.311 Type "Part 3 finds patents according to logical combinations of".
3.312 Type "index terms.".
3.319 Line.
3.32 Demand K as "I want part ".
3.321 Done if K=0.
3.33 To part 115 if K=1.
3.34 To part 116 if K=2.
3.341 To part 119 if K=3.
3.35 Type "Please give a valid part as described above.".
3.36 To step 3.32.

4.00 Use file 174 (ELEC4).
4.05 Set r=0.
4.10 Demand u as "Number of variables used in above expression".
4.12 To step 4.9 if u>12 or u<1.
4.20 Type "For each variable give the corresponding index term.".
4.21 Do step 120 for z=1(1)u if 1≤u≤12.
4.3  Recall item 10.
4.31 Let P be sparse.
4.4 Set m=0.
4.5 Do part 5 for Z=1(1)5.
4.51 Type "No entries found for this search." if m=0.
4.52 Type "Search complete.".
4.6 Use file 174 (ELEC4).
4.7 To part 121.
4.9 Type "Please limit number of variables to 12.".
4.91 To part 119.

5.0 Done if r=1.
5.1 Use file 173 (ELEC3).
5.2 Recall item Z.
5.25 Let A be sparse.
5.3 Set q=-250+100•(Z-1).
5.4 Do part 33 for x=q(1)(q+99).

7 To step 3.32.

33.1 Set r=1 if P(x)=0.
33.2 Done if P(x)=0.
33.25 Set Q=T.
33.27 Type "The following patents have been located:" if Q and m=0.
```

```
33.3 Type P(x) in form 118 if Q.
33.4 Set m=1 if Q.

112.0  Use file 173 (ELEC3).
112.1 Recall item z.
112.9 Use file 174 (ELEC4).

115.02 Recall item 10.
115.03 Set A(0,0)=0.
115.04 Delete A.
115.10 Demand o as "Patent Number".
115.11 To step 3.32 if o=0.
115.12 Set P(250)=o.
115.13 Set y=first(q=-250(1)250:P(q)=o).
115.14 To step 115.8 if y=250.
115.15 Type o in form 115.
115.16 Set m=0.
115.17 Set z=1+ip[(250+y)/100].
115.18 Do part 112.
115.19 Let A be sparse.
115.20 Do part 118 for x=-250(1)250.
115.205 Type "No terms found." if m=0.
115.21 Line.
115.22 To step 115.10.
115.8  Type "This patent number not in file.".
115.81 To step 115.10.

116.01 Set A(0,0) = 0.
116.02 Delete A.
116.10 Demand x as "Index Term".
116.11 To step 3.32 if x=0.
116.12 Type x in form 117.
116.20 Set m=0.
116.210 Use file 173 (ELEC3).
116.211 Set Z=1.
116.212 Set q=-250+100*(Z-1).
116.213 Recall item Z.
116.214 Let A be sparse.
116.215 Do part 117 for y=q(1)(q+99).
116.216 Delete A.
116.217 Set Z=Z+1.
116.218 To step 116.212 if Z<6.
116.219 Use file 174 (ELEC4).
116.22 Type "No entries found for this index term." if m=0.
116.30 To step 116.10.

117.1  Type A(x,y) in form 118 if A(x,y)≠0.
117.2  Set m=1 if A(x,y)≠0.

118.10 Type x in form 116 if A(x,y)=o.
118.20 Set m=1 if A(x,y)=o.

119.0 Line.
119.1 Type "On next line type a search expression.  Then type: Do part 4.".
```

```
119.3 Line.

120.  Do step (120+z/100).
120.01 Demand t(1) as "a".
120.02 Demand t(2) as "b".
120.03 Demand t(3) as "c".
120.04 Demand t(4) as "d".
120.05 Demand t(5) as "e".
120.06 Demand t(6) as "f".
120.07 Demand t(7) as "g".
120.08 Demand t(8) as "h".
120.09 Demand t(9) as "i".
120.10 Demand t(10) as "j".
120.11 Demand t(11) as "k".
120.12 Demand t(12) as "l".

121.0 Line.
121.1 Type "Enter a new search expression if you wish. Then do part 4.".
121.2 Line.

Form 115:
Patent number _____ has the following index terms:

Form 116:

____

Form 117:
The following patents refer to index term ____:

Form 118:

_______

          a:  w(1)
          b:  w(2)
          c:  w(3)
          d:  w(4)
          e:  w(5)
          f:  w(6)
          g:  w(7)
          h:  w(8)
          i:  w(9)
          j:  w(10)
          k:  w(11)
          l:  w(12)
       w(y):  tv(A(t(y),x))
```